IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

System and Method for Distributed Management of Shared Computers

Inventor(s):

Galen C. Hunt
Aamer Hydrie
Steven P. Levi
Jakob Rehof
David S. Stutz
Bassam Tabbara
Mark D. VanAntwerp
Robert V. Welland

ATTORNEY'S DOCKET NO. MS1-547US

TECHNICAL FIELD

This invention relates to computer system management. More particularly, the invention relates to the distributed management of shared computers.

BACKGROUND OF THE INVENTION

The Internet and its use have expanded greatly in recent years, and this expansion is expected to continue. One significant way in which the Internet is used is the World Wide Web (also referred to as the "web"), which is a collection of documents (referred to as "web pages") that users can view or otherwise render and which typically include links to one or more other pages that the user can access. Many businesses and individuals have created a presence on the web, typically consisting of one or more web pages describing themselves, describing their products or services, identifying other information of interest, allowing goods or services to be purchased, etc.

Web pages are typically made available on the web via one or more web servers, a process referred to as "hosting" the web pages. Sometimes these web pages are freely available to anyone that requests to view them (e.g., a company's advertisements) and other times access to the web pages is restricted (e.g., a password may be necessary to access the web pages). Given the large number of people that may be requesting to view the web pages (especially in light of the global accessibility to the web), a large number of servers may be necessary to adequately host the web pages (e.g., the same web page can be hosted on multiple servers to increase the number of people that can access the web page concurrently). Additionally, because the web is geographically distributed and has non-uniformity of access, it is often desirable to distribute servers to diverse

remote locations in order to minimize access times for people in diverse locations of the world. Furthermore, people tend to view web pages around the clock (again, especially in light of the global accessibility to the web), so servers hosting web pages should be kept functional 24 hours per day.

Managing a large number of servers, however, can be difficult. A reliable power supply is necessary to ensure the servers can run. Physical security is necessary to ensure that a thief or other mischievous person does not attempt to damage or steal the servers. A reliable Internet connection is required to ensure that the access requests will reach the servers. A proper operating environment (e.g., temperature, humidity, etc.) is required to ensure that the servers operate properly. Thus, "co-location facilities" have evolved which assist companies in handling these difficulties.

A co-location facility refers to a complex that can house multiple servers. The co-location facility typically provides a reliable Internet connection, a reliable power supply, and proper operating environment. The co-location facility also typically includes multiple secure areas (e.g., cages) into which different companies can situate their servers. The collection of servers that a particular company situates at the co-location facility is referred to as a "server cluster", even though in fact there may only be a single server at any individual co-location facility. The particular company is then responsible for managing the operation of the servers in their server cluster.

Such co-location facilities, however, also present problems. One problem is data security. Different companies (even competitors) can have server clusters at the same co-location facility. Care is required, in such circumstances, to ensure that data received from the Internet (or sent by a server in the server cluster) that is

intended for one company is not routed to a server of another company situated at the co-location facility.

An additional problem is the management of the servers once they are placed in the co-location facility. Currently, a system administrator from a company is able to contact a co-location facility administrator (typically by telephone) and ask him or her to reset a particular server (typically by pressing a hardware reset button on the server, or powering off then powering on the server) in the event of a failure of (or other problem with) the server. This limited reset-only ability provides very little management functionality to the company. Alternatively, the system administrator from the company can physically travel to the co-location facility him/her-self and attend to the faulty server. Unfortunately, a significant amount of time can be wasted by the system administrator in traveling to the co-location facility to attend to a server. Thus, it would be beneficial to have an improved way to manage remote server computers at a co-location facility.

Another problem concerns the enforcement of the rights of both the operators of the servers in the co-location facility and the operators of the web service hosted on those servers. The operators of the servers need to be able to maintain their rights (e.g., re-possessing areas of the facility where servers are stored), even though the servers are owned by the operators of the web service. Additionally, the operators of the web service need to be assured that their data remains secure.

The invention described below addresses these disadvantages, improving the distributed management of shared computers in co-location facilities.

SUMMARY OF THE INVENTION

Distributed management of shared computers is described herein.

According to one aspect, a multi-tiered management architecture is employed including an application development tier, an application operations tier, and a cluster operations tier. In the application development tier, applications are developed for execution on one or more server computers. In the application operations tier, execution of the applications is managed and sub-boundaries within a cluster of servers at a co-location facility may be established. In the cluster operations tier, operation of the server computers is managed without concern for what applications are executing on the one or more server computers, and server cluster boundaries at the co-location facility may be established.

According to another aspect, a co-location facility includes multiple server clusters, each corresponding to a different customer. For each server cluster, a cluster operations management console is implemented locally at the co-location facility to manage hardware operations of the cluster, and an application operations management console is implemented at a location remote from the co-location facility to manage software operations of the cluster. In the event of a hardware failure, the cluster operations management console takes corrective action (e.g., notifying an administrator at the co-location facility or attempting to correct the failure itself). In the event of a software failure, the application operations management console takes corrective action (e.g., notifying one of the customer's administrators or attempting to correct the failure itself).

According to another aspect, boundaries of a server cluster are established by a cluster operations management console. Establishment of the boundaries ensures that data is routed only to nodes within the server cluster, and not to other

nodes at the co-location facility that are not part of the server cluster. Further sub-boundaries within a server cluster may be established by an application operations management console to ensure data is routed only to particular nodes within the server cluster.

According to another aspect, rights to multiple server computers to be located at a co-location facility are sold to a customer and a multiple-tiered management scheme is enforced on the server computers. According to the multiple-tiered management scheme, hardware operation of the server computers is managed locally at the co-location facility whereas software operation of the server computers is managed from a location remote from the co-location facility. The server computers can be either sold to the customer or leased to the customer.

According to another aspect, a landlord/tenant relationship is created using one or more server computers at a co-location facility. The operator of the co-location facility supplies the facility as well as the servers (and thus can be viewed as a "landlord"), while customers of the facility lease the use of the facility as well as servers at that facility (and thus can be viewed as "tenants"). This landlord/tenant relationship allows the landlord to establish clusters of computers for different tenants and establish boundaries between clusters so that a tenant's data does not pass beyond its cluster (and to another tenant's cluster). Additionally, encryption is employed in various manners to assure the tenant that information stored at the servers it leases cannot be viewed by anyone else, even if the tenant terminates its lease or returns to the landlord one of the servers it is leasing.

According to another aspect, a multi-tiered management architecture is employed in managing computers that are not part of a co-location facility. This

multi-tiered architecture is used for managing computers (whether server computers or otherwise) in a variety of settings, such as businesses, homes, etc.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example and not limitation in the figures of the accompanying drawings. The same numbers are used throughout the figures to reference like components and/or features.

Fig. 1 shows a client/server network system and environment such as may be used with certain embodiments of the invention.

Fig. 2 shows a general example of a computer that can be used in accordance with certain embodiments of the invention.

Fig. 3 is a block diagram illustrating an exemplary co-location facility in more detail.

Fig. 4 is a block diagram illustrating an exemplary multi-tiered management architecture.

Fig. 5 is a block diagram illustrating an exemplary node in more detail in accordance with certain embodiments of the invention.

Fig. 6 is a flowchart illustrating an exemplary process for encryption key generation and distribution in accordance with certain embodiments of the invention.

Fig. 7 is a flowchart illustrating an exemplary process for the operation of a cluster operations management console in accordance with certain embodiments of the invention.

Fig. 8 is a flowchart illustrating an exemplary process for the operation of an application operations management console in accordance with certain embodiments of the invention.

DETAILED DESCRIPTION

Fig. 1 shows a client/server network system and environment such as may be used with certain embodiments of the invention. Generally, the system includes multiple (*n*) client computers 102 and multiple (*m*) co-location facilities 104 each including multiple clusters of server computers (server clusters) 106. The servers and client computers communicate with each other over a data communications network 108. The communications network in Fig. 1 comprises a public network 108 such as the Internet. Other types of communications networks might also be used, in addition to or in place of the Internet, including local area networks (LANs), wide area networks (WANs), etc. Data communications network 108 can be implemented in any of a variety of different manners, including wired and/or wireless communications media.

Communication over network 108 can be carried out using any of a wide variety of communications protocols. In one implementation, client computers 102 and server computers in clusters 106 can communicate with one another using the Hypertext Transfer Protocol (HTTP), in which web pages are hosted by the server computers and written in a markup language, such as the Hypertext Markup Language (HTML) or the eXtensible Markup Language (XML).

In the discussions herein, embodiments of the invention are described primarily with reference to implementation at a co-location facility (such as facility 104). The invention, however, is not limited to such implementations and

can be used for distributed management in any of a wide variety of situations. For example, in situations where all of the servers at a facility are owned or leased to the same customer, in situations where a single computing device (e.g., a server or client) is being managed, in situations where computers (whether servers or otherwise) in a business or home environment are being managed, etc.

In the discussion herein, embodiments of the invention are described in the general context of computer-executable instructions, such as program modules, being executed by one or more conventional personal computers. Generally, program modules include routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. Moreover, those skilled in the art will appreciate that various embodiments of the invention may be practiced with other computer system configurations, including hand-held devices, gaming consoles, Internet appliances, multiprocessor systems, microprocessor-based or programmable consumer electronics, network PCs, minicomputers, mainframe computers, and the like. In a distributed computer environment, program modules may be located in both local and remote memory storage devices.

Alternatively, embodiments of the invention can be implemented in hardware or a combination of hardware, software, and/or firmware. For example, all or part of the invention can be implemented in one or more application specific integrated circuits (ASICs) or programmable logic devices (PLDs).

Fig. 2 shows a general example of a computer 142 that can be used in accordance with certain embodiments of the invention. Computer 142 is shown as an example of a computer that can perform the functions of a client computer 102 of Fig. 1, a computer or node in a co-location facility 104 of Fig. 1 or other

location (e.g., node 248 of Fig. 5 below), or a local or remote management console as discussed in more detail below.

Computer 142 includes one or more processors or processing units 144, a system memory 146, and a bus 148 that couples various system components including the system memory 146 to processors 144. The bus 148 represents one or more of any of several types of bus structures, including a memory bus or memory controller, a peripheral bus, an accelerated graphics port, and a processor or local bus using any of a variety of bus architectures. The system memory includes read only memory (ROM) 150 and random access memory (RAM) 152. A basic input/output system (BIOS) 154, containing the basic routines that help to transfer information between elements within computer 142, such as during start-up, is stored in ROM 150.

Computer 142 further includes a hard disk drive 156 for reading from and writing to a hard disk, not shown, connected to bus 148 via a hard disk driver interface 157 (e.g., a SCSI, ATA, or other type of interface); a magnetic disk drive 158 for reading from and writing to a removable magnetic disk 160, connected to bus 148 via a magnetic disk drive interface 161; and an optical disk drive 162 for reading from or writing to a removable optical disk 164 such as a CD ROM, DVD, or other optical media, connected to bus 148 via an optical drive interface 165. The drives and their associated computer-readable media provide nonvolatile storage of computer readable instructions, data structures, program modules and other data for computer 142. Although the exemplary environment described herein employs a hard disk, a removable magnetic disk 160 and a removable optical disk 164, it should be appreciated by those skilled in the art that other types of computer readable media which can store data that is accessible by a computer,

such as magnetic cassettes, flash memory cards, digital video disks, random access memories (RAMs) read only memories (ROM), and the like, may also be used in the exemplary operating environment.

A number of program modules may be stored on the hard disk, magnetic disk 160, optical disk 164, ROM 150, or RAM 152, including an operating system 170, one or more application programs 172, other program modules 174, and program data 176. A user may enter commands and information into computer 142 through input devices such as keyboard 178 and pointing device 180. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are connected to the processing unit 144 through an interface 168 that is coupled to the system bus. A monitor 184 or other type of display device is also connected to the system bus 148 via an interface, such as a video adapter 186. In addition to the monitor, personal computers typically include other peripheral output devices (not shown) such as speakers and printers.

Computer 142 optionally operates in a networked environment using logical connections to one or more remote computers, such as a remote computer 188. The remote computer 188 may be another personal computer, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to computer 142, although only a memory storage device 190 has been illustrated in Fig. 2. The logical connections depicted in Fig. 2 include a local area network (LAN) 192 and a wide area network (WAN) 194. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets, and the Internet. In the described embodiment of the invention, remote computer 188

executes an Internet Web browser program (which may optionally be integrated into the operating system 170) such as the "Internet Explorer" Web browser manufactured and distributed by Microsoft Corporation of Redmond, Washington.

When used in a LAN networking environment, computer 142 is connected to the local network 192 through a network interface or adapter 196. When used in a WAN networking environment, computer 142 typically includes a modem 198 or other component for establishing communications over the wide area network 194, such as the Internet. The modem 198, which may be internal or external, is connected to the system bus 148 via an interface (e.g., a serial port interface 168). In a networked environment, program modules depicted relative to the personal computer 142, or portions thereof, may be stored in the remote memory storage device. It is to be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

Generally, the data processors of computer 142 are programmed by means of instructions stored at different times in the various computer-readable storage media of the computer. Programs and operating systems are typically distributed, for example, on floppy disks or CD-ROMs. From there, they are installed or loaded into the secondary memory of a computer. At execution, they are loaded at least partially into the computer's primary electronic memory. The invention described herein includes these and other various types of computer-readable storage media when such media contain instructions or programs for implementing the steps described below in conjunction with a microprocessor or other data processor. The invention also includes the computer itself when programmed according to the methods and techniques described below. Furthermore, certain

sub-components of the computer may be programmed to perform the functions and steps described below. The invention includes such sub-components when they are programmed as described. In addition, the invention described herein includes data structures, described below, as embodied on various types of memory media.

For purposes of illustration, programs and other executable program components such as the operating system are illustrated herein as discrete blocks, although it is recognized that such programs and components reside at various times in different storage components of the computer, and are executed by the data processor(s) of the computer.

Fig. 3 is a block diagram illustrating an exemplary co-location facility in more detail. Co-location facility 104 is illustrated including multiple nodes (also referred to as server computers) 210. Co-location facility 104 can include any number of nodes 210, and can easily include an amount of nodes numbering into the thousands.

The nodes 210 are grouped together in clusters, referred to as server clusters (or node clusters). For ease of explanation and to avoid cluttering the drawings, only a single cluster 212 is illustrated in Fig. 3. Each server cluster includes nodes 210 that correspond to a particular customer of co-location facility 104. The nodes 210 of a server cluster are physically isolated from the nodes 210 of other server clusters. This physical isolation can take different forms, such as separate locked cages or separate rooms at co-location facility 104. Physically isolating server clusters ensures customers of co-location facility 104 that only they can physically access their nodes (other customers cannot). Alternatively,

server clusters may be logically, but not physically, isolated for each other (e.g., using cluster boundaries as discussed in more detail below).

A landlord/tenant relationship (also referred to as a lessor/lessee relationship) can also be established based on the nodes 210. The owner (and/or operator) of co-location facility 104 owns (or otherwise has rights to) the individual nodes 210, and thus can be viewed as a "landlord". The customers of co-location facility 104 lease the nodes 210 from the landlord, and thus can be viewed as a "tenant". The landlord is typically not concerned with what types of data or programs are being stored at the nodes 210 by the tenant, but does impose boundaries on the clusters that prevent nodes 210 from different clusters from communicating with one another, as discussed in more detail below.

The landlord/tenant relationship is discussed herein primarily with reference to only two levels: the landlord and the tenant. However, in alternate embodiments this relationship can be expanded to any number of levels. For example, the landlord may share its management responsibilities with one or more sub-landlords (each of which would have certain managerial control over one or more nodes 210), and the tenant may similarly share its management responsibilities with one or more sub-tenants (each of which would have certain managerial control over one or more nodes 210).

Although physically isolated, nodes 210 of different clusters are often physically coupled to the same transport medium (or media) 211 that enables access to network connection(s) 216, and possibly application operations management console 242, discussed in more detail below. This transport medium can be wired or wireless.

As each node 210 can be coupled to a shared transport medium 211, each node 210 is configurable to restrict which other nodes 210 data can be sent to or received from. Given that a number of different nodes 210 may be included in a tenant's server cluster, the tenant may want to be able to pass data between different nodes 210 within the cluster for processing, storage, etc. However, the tenant will typically not want data to be passed to other nodes 210 that are not in the server cluster. Configuring each node 210 in the cluster to restrict which other nodes 210 data can be sent to or received from allows a boundary for the server cluster to be established and enforced. Establishment and enforcement of such server cluster boundaries prevents tenant data from being erroneously or improperly forwarded to a node that is not part of the cluster.

These initial boundaries established by the landlord prevent communication between nodes 210 of different tenants, thereby ensuring that each tenant's data can be passed to other nodes 210 of that tenant. The tenant itself may also further define sub-boundaries within its cluster, establishing sub-clusters of nodes 210 that data cannot be communicated out of (or in to) either to or from other nodes in the cluster. The tenant is able to add, modify, remove, etc. such sub-cluster boundaries at will, but only within the boundaries defined by the landlord (that is, the cluster boundaries). Thus, the tenant is not able to alter boundaries in a manner that would allow communication to or from a node 210 to extend to another node 210 that is not within the same cluster.

Co-location facility 104 supplies reliable power 214 and reliable network connection(s) 216 to each of the nodes 210. Power 214 and network connection(s) 216 are shared by all of the nodes 210, although alternatively separate power 214 and network connection(s) 216 may be supplied to nodes 210 or groupings (e.g.,

clusters) of nodes. Any of a wide variety of conventional mechanisms for supplying reliable power can be used to supply reliable power 214, such as power received from a public utility company along with backup generators in the event of power failures, redundant generators, batteries, fuel cells, or other power storage mechanisms, etc. Similarly, any of a wide variety of conventional mechanisms for supplying a reliable network connection can be used to supply network connection(s) 216, such as redundant connection transport media, different types of connection media, different access points (e.g., different Internet access points, different Internet service providers (ISPs), etc.).

In certain embodiments, nodes 210 are leased or sold to customers by the operator or owner of co-location facility 104 along with the space (e.g., locked cages) and service (e.g., access to reliable power 214 and network connection(s) 216) at facility 104. In other embodiments, space and service at facility 104 may be leased to customers while one or more nodes are supplied by the customer.

Management of each node 210 is carried out in a multiple-tiered manner. Fig. 4 is a block diagram illustrating an exemplary multi-tiered management architecture. The multi-tiered architecture includes three tiers: a cluster operations management tier 230, an application operations management tier 232, and an application development tier 234. Cluster operations management tier 230 is implemented locally at the same location as the server(s) being managed (e.g., at a co-location facility) and involves managing the hardware operations of the server(s). In the illustrated example, cluster operations management tier 230 is not concerned with what software components are executing on the nodes 210, but only with the continuing operation of the hardware of nodes 210 and establishing any boundaries between clusters of nodes.

The application operations management tier 232, on the other hand, is implemented at a remote location other than where the server(s) being managed are located (e.g., other than the co-location facility), but from a client computer that is still communicatively coupled to the server(s). The application operations management tier 232 involves managing the software operations of the server(s) and defining sub-boundaries within server clusters. The client can be coupled to the server(s) in any of a variety of manners, such as via the Internet or via a dedicated (e.g., dial-up) connection. The client can be coupled continually to the server(s), or alternatively sporadically (e.g., only when needed for management purposes).

The application development tier 234 is implemented on another client computer at a location other than the server(s) (e.g., other than at the co-location facility) and involves development of software components or engines for execution on the server(s). Alternatively, current software on a node 210 at co-location facility 104 could be accessed by a remote client to develop additional software components or engines for the node. Although the client at which application development tier 234 is implemented is typically a different client than that at which application operations management tier 232 is implemented, tiers 232 and 234 could be implemented (at least in part) on the same client.

Although only three tiers are illustrated in Fig. 4, alternatively the multi-tiered architecture could include different numbers of tiers. For example, the application operations management tier may be separated into two tiers, each having different (or overlapping) responsibilities, resulting in a 4-tiered architecture. The management at these tiers may occur from the same place (e.g., a single application operations management console may be shared), or

alternatively from different places (e.g., two different operations management consoles).

Returning to Fig. 3, co-location facility 104 includes a cluster operations management console for each server cluster. In the example of Fig. 3, cluster operations management console 240 corresponds to cluster 212. Cluster operations management console 240 implements cluster operations management tier 230 (Fig. 4) for cluster 212 and is responsible for managing the hardware operations of nodes 210 in cluster 212. Cluster operations management console 240 monitors the hardware in cluster 212 and attempts to identify hardware failures. Any of a wide variety of hardware failures can be monitored for, such as processor failures, bus failures, memory failures, etc. Hardware operations can be monitored in any of a variety of manners, such as cluster operations management console 240 sending test messages or control signals to the nodes 210 that require the use of particular hardware in order to respond (no response or an incorrect response indicates failure), having messages or control signals that require the use of particular hardware to generate periodically sent by nodes 210 to cluster operations management console 240 (not receiving such a message or control signal within a specified amount of time indicates failure), etc. Alternatively, cluster operations management console 240 may make no attempt to identify what type of hardware failure has occurred, but rather simply that a failure has occurred.

Once a hardware failure is detected, cluster operations management console 240 acts to correct the failure. The action taken by cluster operations management console 240 can vary based on the hardware as well as the type of failure, and can vary for different server clusters. The corrective action can be notification of an administrator (e.g., a flashing light, an audio alarm, an electronic mail message,

calling a cell phone or pager, etc.), or an attempt to physically correct the problem (e.g., reboot the node, activate another backup node to take its place, etc.).

Cluster operations management console 240 also establishes cluster boundaries within co-location facility 104. The cluster boundaries established by console 240 prevent nodes 210 in one cluster (e.g., cluster 212) from communicating with nodes in another cluster (e.g., any node not in cluster 212), while at the same time not interfering with the ability of nodes 210 within a cluster from communicating with other nodes within that cluster. These boundaries provide security for the tenants' data, allowing them to know that their data cannot be communicated to other tenants' nodes 210 at facility 104 even though network connection 216 may be shared by the tenants.

In the illustrated example, each cluster of co-location facility 104 includes a dedicated cluster operations management console. Alternatively, a single cluster operations management console may correspond to, and manage hardware operations of, multiple server clusters. According to another alternative, multiple cluster operations management consoles may correspond to, and manage hardware operations of, a single server cluster. Such multiple consoles can manage a single server cluster in a shared manner, or one console may operate as a backup for another console (e.g., providing increased reliability through redundancy, to allow for maintenance, etc.).

An application operations management console 242 is also communicatively coupled to co-location facility 104. Application operations management console 242 is located at a location remote from co-location facility 104 (that is, not within co-location facility 104), typically being located at the offices of the customer. A different application operations management console

242 corresponds to each server cluster of co-location facility 104, although alternatively multiple consoles 242 may correspond to a single server cluster, or a single console 242 may correspond to multiple server clusters. Application operations management console 242 implements application operations management tier 232 (Fig. 4) for cluster 212 and is responsible for managing the software operations of nodes 210 in cluster 212 as well as securing sub-boundaries within cluster 212.

Application operations management console 242 monitors the software in cluster 212 and attempts to identify software failures. Any of a wide variety of software failures can be monitored for, such as application processes or threads that are "hung" or otherwise non-responsive, an error in execution of application processes or threads, etc. Software operations can be monitored in any of a variety of manners (similar to the monitoring of hardware operations discussed above), such as application operations management console 242 sending test messages or control signals to particular processes or threads executing on the nodes 210 that require the use of particular routines in order to respond (no response or an incorrect response indicates failure), having messages or control signals that require the use of particular software routines to generate periodically sent by processes or threads executing on nodes 210 to application operations management console 242 (not receiving such a message or control signal within a specified amount of time indicates failure), etc. Alternatively, application operations management console 242 may make no attempt to identify what type of software failure has occurred, but rather simply that a failure has occurred.

Once a software failure is detected, application operations management console 242 acts to correct the failure. The action taken by application operations

management console 242 can vary based on the hardware as well as the type of failure, and can vary for different server clusters. The corrective action can be notification of an administrator (e.g., a flashing light, an audio alarm, an electronic mail message, calling a cell phone or pager, etc.), or an attempt to correct the problem (e.g., reboot the node, re-load the software component or engine image, terminate and re-execute the process, etc.).

Thus, the management of a node 210 is distributed across multiple managers, regardless of the number of other nodes (if any) situated at the same location as the node 210. The multi-tiered management allows the hardware operations management to be separated from the application operations management, allowing two different consoles (each under the control of a different entity) to share the management responsibility for the node.

The multi-tiered management architecture can also be used in other situations to manage one or more computers from one or more remote locations, even if the computers are not part of a co-location facility. By way of example, a small business may purchase their own computers, but hire another company to manage the hardware operations of the computers, and possibly yet another company to manage the software operations of the computers.

In this example, the small business (the owner of the computers) is a first management tier. The owner then leases the computers to the outsourced hardware operator, which is the second management tier. The hardware operator can manage the hardware operation from a control console, either located locally at the small business along with the computers being managed or alternatively at some remote location, analogous to cluster operations management console 240. The hardware operator then leases the computers to an outsourced software

operator, which is the third management tier. The software operator can manage the software operation from a control console, either located locally at the small business along with the computers being managed or alternatively at some remote location, analogous to application operations management console 242. The software operator then leases the computers back to their owner, so the owner becomes the "user" of the computers, which is the fourth management tier. During normal operation, the computer owner occupies this fourth management tier. However, the computer owner can exercise its first management tier rights to sever one or both of the leases to the software operator and the hardware operator, such as when the computer owner desires to change software or hardware operators.

Fig. 5 is a block diagram illustrating an exemplary node in more detail in accordance with certain embodiments of the invention. Node 248 is an exemplary node managed by other devices (e.g., consoles 240 and 242 of Fig. 3) external to the node. Node 248 can be a node 210 of Fig. 3, or alternatively a node at another location (e.g., a computer in a business or home environment). Node 248 includes a monitor 250, referred to as the "BMonitor", and a plurality of software components or engines 252, and is coupled to (or alternatively incorporates) a mass storage device 262. In the illustrated example, node 248 is a server computer having a processor(s) that supports multiple privilege levels (e.g., rings in an x86 architecture processor). In the illustrated example, these privilege levels are referred to as rings, although alternate implementations using different processor architectures may use different nomenclature. The multiple rings provide a set of prioritized levels that software can execute at, often including 4 levels (Rings 0, 1, 2, and 3). Ring 0 is typically referred to as the most privileged ring. Software

processes executing in Ring 0 can typically access more features (e.g., instructions) than processes executing in less privileged Rings. Furthermore, a processor executing in a particular Ring cannot alter code or data in a higher priority ring. In the illustrated example, BMonitor 250 executes in Ring 0, while engines 252 execute in Ring 1 (or alternatively Rings 2 and/or 3). Thus, the code or data of BMonitor 250 (executing in Ring 0) cannot be altered directly by engines 252 (executing in Ring 1). Rather, any such alterations would have to be made by an engine 252 requesting BMonitor 250 to make the alteration (e.g., by sending a message to BMonitor 250, invoking a function of BMonitor 250, etc.). Implementing BMonitor 250 in Ring 0 protects BMonitor 250 from a rogue or malicious engine 252 that tries to bypass any restrictions imposed by BMonitor 250.

BMonitor 250 is the fundamental control module of node 248 – it controls (and optionally includes) both the network interface card and the memory manager. By controlling the network interface card (which may be separate from BMonitor 250, or alternatively BMonitor 250 may be incorporated on the network interface card), BMonitor 250 can control data received by and sent by node 248. By controlling the memory manager, BMonitor 250 controls the allocation of memory to engines 252 executing in node 248 and thus can assist in preventing rogue or malicious engines from interfering with the operation of BMonitor 250.

Although various aspects of node 248 may be under control of BMonitor 250 (e.g., the network interface card), BMonitor 250 still makes at least part of such functionality available to engines 252 executing on the node 248. BMonitor 250 provides an interface (e.g., via controller 254 discussed in more detail below) via which engines 252 can request access to the functionality, such as to send data

out to another node 248 or to the Internet. These requests can take any of a variety of forms, such as sending messages, calling a function, etc.

BMonitor 250 includes controller 254, network interface 256, one or more filters 258, and a Distributed Host Control Protocol (DHCP) module 260. Network interface 256 provides the interface between node 248 and the network (e.g., network connections 126 of Fig. 3) via the internal transport medium 211 of co-location facility 104. Filters 258 identify other nodes 248 (and/or other sources or targets (e.g., coupled to Internet 108 of Fig. 1) that data can (or alternatively cannot) be sent to and/or received from. The nodes or other sources/targets can be identified in any of a wide variety of manners, such as by network address (e.g., Internet Protocol (IP) address), some other globally unique identifier, a locally unique identifier (e.g., a numbering scheme proprietary or local to co-location facility 104), etc.

Filters 258 can fully restrict access to a node (e.g., no data can be received from or sent to the node), or partially restrict access to a node. Partial access restriction can take different forms. For example, a node may be restricted so that data can be received from the node but not sent to the node (or vice versa). By way of another example, a node may be restricted so that only certain types of data (e.g., communications in accordance with certain protocols, such as HTTP) can be received from and/or sent to the node. Filtering based on particular types of data can be implemented in different manners, such as by communicating data in packets with header information that indicate the type of data included in the packet.

Filters 258 can be added by application operations management console 242 or cluster operations management console 240. In the illustrated example,

filters added by cluster operations management console 240 (to establish cluster boundaries) restrict full access to nodes (e.g., any access to another node can be prevented) whereas filters added by application operations management console 242 (to establish sub-boundaries within a cluster) can restrict either full access to nodes or partial access.

Controller 254 also imposes some restrictions on what filters can be added to filters 258. In the illustrated example, controller 254 allows cluster operations management console 240 to add any filters it desires (which will define the boundaries of the cluster). However, controller 254 restricts application operations management console 242 to adding only filters that are at least as restrictive as those added by console 240. If console 242 attempts to add a filter that is less restrictive than those added by console 240 (in which case the sub-boundary may extend beyond the cluster boundaries), controller 254 refuses to add the filter (or alternatively may modify the filter so that it is not less restrictive). By imposing such a restriction, controller 254 can ensure that the sub-boundaries established at the application operations management level do not extend beyond the cluster boundaries established at the cluster operations management level.

Controller 254, using one or more filters 258, operates to restrict data packets sent from node 248 and/or received by node 248. All data intended for an engine 252, or sent by an engine 252, to another node, is passed through network interface 256 and filters 258. Controller 254 applies the filters 258 to the data, comparing the target of the data (e.g., typically identified in a header portion of a packet including the data) to acceptable (and/or restricted) nodes (and/or network addresses) identified in filters 258. If filters 258 indicate that the target of the data is acceptable, then controller 254 allows the data to pass through to the target

(either into node 248 or out from node 248). However, if filters 258 indicate that the target of the data is not acceptable, then controller 254 prevents the data from passing through to the target. Controller 254 may return an indication to the source of the data that the data cannot be passed to the target, or may simply ignore or discard the data.

The application of filters 258 to the data by controller 254 allows the boundary restrictions of a server cluster to be imposed. Filters 258 can be programmed (e.g., by application operations management console 242 of Fig. 3) with the node addresses of all the nodes within the server cluster (e.g., cluster 212). Controller 254 then prevents data received from any node not within the server cluster from being passed through to an engine 252, and similarly prevents any data being sent to a node other than one within the server cluster from being sent. Similarly, data received from Internet 108 (Fig. 1) can identify a target node 210 (e.g., by IP address), so that controller 254 of any node other than the target node will prevent the data from being passed through to an engine 252.

DHCP module 260 implements the Distributed Host Control Protocol, allowing BMonitor 250 (and thus node 210) to obtain an IP address from a DHCP server (e.g., cluster operations management console 240 of Fig. 3). During an initialization process for node 210, DHCP module 260 requests an IP address from the DHCP server, which in turn provides the IP address to module 260. Additional information regarding DHCP is available from Microsoft Corporation of Redmond, Washington.

Software engines 252 include any of a wide variety of conventional software components. Examples of engines 252 include an operating system (e.g., Windows NT®), a load balancing server component (e.g., to balance the

processing load of multiple nodes 248), a caching server component (e.g., to cache data and/or instructions from another node 248 or received via the Internet), a storage manager component (e.g., to manage storage of data from another node 248 or received via the Internet), etc. In one implementation, each of the engines 252 is a protocol-based engine, communicating with BMonitor 250 and other engines 252 via messages and/or function calls without requiring the engines 252 and BMonitor 250 to be written using the same programming language.

Controller 254 is further responsible for controlling the execution of engines 252. This control can take different forms, including beginning execution of an engine 252, terminating execution of an engine 252, re-loading an image of an engine 252 from a storage device, debugging execution of an engine 252, etc. Controller 254 receives instructions from application operations management console 242 of Fig. 3 regarding which of these control actions to take and when to take them. Thus, the control of engines 252 is actually managed by the remote application operations management console 242, not locally at co-location facility 104. Controller 254 also provides an interface via which application operations management console 242 can identify filters to add (and/or remove) from filter set 258.

Controller 254 also includes an interface via which cluster operations management console 240 of Fig. 3 can communicate commands to controller 254. Different types of hardware operation oriented commands can be communicated to controller 254 by cluster operations management console 240, such as re-booting the node, shutting down the node, placing the node in a low-power state (e.g., in a suspend or standby state), changing cluster boundaries, changing encryption keys, etc.

Controller 254 further provides encryption support for BMonitor 250, allowing data to be stored securely on mass storage device 262 (e.g., a magnetic disk, an optical disk, etc.) and secure communications to occur between node 248 and an operations management console (e.g., console 240 or 242 of Fig. 3). Controller 254 maintains multiple encryption keys, including: one for the landlord (referred to as the "landlord key") which accesses node 248 from cluster operations management console 240, one for the lessee of node 248 (referred to as the "tenant key") which accesses node 248 from application operations management console 242, and keys that BMonitor 250 uses to securely store data on mass storage device 262 (referred to as the "disk key").

BMonitor 250 makes use of public key cryptography to provide secure communications between node 248 and the management consoles (e.g., consoles 240 and 242). Public key cryptography is based on a key pair, including both a public key and a private key, and an encryption algorithm. The encryption algorithm can encrypt data based on the public key such that it cannot be decrypted efficiently without the private key. Thus, communications from the public-key holder can be encrypted using the public key, allowing only the private-key holder to decrypt the communications. Any of a variety of public key cryptography techniques may be used, such as the well-known RSA (Rivest, Shamir, and Adelman) encryption technique. For a basic introduction of cryptography, the reader is directed to a text written by Bruce Schneier and entitled "Applied Cryptography: Protocols, Algorithms, and Source Code in C," published by John Wiley & Sons with copyright 1994 (or second edition with copyright 1996).

BMonitor 250 is initialized to include a public/private key pair for both the landlord and the tenant. These key pairs can be generated by BMonitor 250, or alternatively by some other component and stored within BMonitor 250 (with that other component being trusted to destroy its knowledge of the key pair). As used herein, U refers to a public key and R refers to a private key. The public/private key pair 264 for the landlord is referred to as (U_L, R_L), and the public/private key pair 266 for the tenant is referred to as (U_T, R_T). BMonitor 250 makes the public keys U_L and U_T available to the landlord, but keeps the private keys R_L and R_T secret. In the illustrated example, BMonitor 250 never divulges the private keys R_L and R_T, so both the landlord and the tenant can be assured that no entity other than the BMonitor 250 can decrypt information that they encrypt using their public keys (e.g., via cluster operations management console 240 and application operations management console 242 of Fig. 3, respectively).

Once the landlord has the public keys U_L and U_T, the landlord can assign node 210 to a particular tenant, giving that tenant the public key U_T. Use of the public key U_T allows the tenant to encrypt communications to BMonitor 250 that only BMonitor 250 can decrypt (using the private key R_T). Although not required, a prudent initial step for the tenant is to request that BMonitor 250 generate a new public/private key pair (U_T, R_T). In response to such a request, a key generator 268 of BMonitor 250 generates a new public/private key pair in any of a variety of well-known manners, stores the new key pair as key pair 266, and returns the new public key U_T to the tenant. By generating a new key pair, the tenant is assured that no other entity, including the landlord, is aware of the tenant public key U_T. Additionally, the tenant may also have new key pairs generated at subsequent times.

BMonitor 250 enforces restrictions on what entities can request new public/private key pairs. The tenant is able to request new tenant public/private key pairs, but is not able to request new landlord public/private key pairs. The landlord, however, can request new landlord public/private key pairs as well as new tenant public/private key pairs. Whenever a request for a new public/private key pair is received, controller 254 verifies the identity of the requestor as the tenant or landlord (e.g., based on a remote log-in procedure, password verification, manner in which the requestor is communicating with or is coupled to node 248, etc.) before generating the new key pair.

In order to ensure bi-directional communication security between BMonitor 250 and the landlord and tenant control devices (e.g., operations management consoles 240 and 242, respectively), the landlord and tenant control devices may also generate (or otherwise be assigned) public/private key pairs. In this situation, consoles 240 and 242 can communicate their respective public keys to BMonitors 250 of nodes 248 they desire (or expect to desire) to communicate with securely. Once the public key of a console is known by a BMonitor 250, the BMonitor 250 can encrypt communications to that console using its public key, thereby preventing any other device except the console having the private key from reading the communication.

BMonitor 250 also maintains a disk key 270, which is generated based on one or more symmetric keys 272 and 274 (symmetric keys refer to secret keys used in secret key cryptography). Disk key 270, also a symmetric key, is used by BMonitor 250 to store information in mass storage device 262. BMonitor 250 keeps disk key 270 secure, using it only to encrypt data node 248 stores on mass storage device 262 and decrypt data node 248 retrieves from mass storage device

262 (thus there is no need for any other entities, including the landlord and tenant, to have knowledge of disk key 270). Alternatively, the landlord or tenant may be informed of disk key 270, or another key on which disk key 270 is based.

Use of disk key 270 ensures that data stored on mass storage device 262 can only be decrypted by the node 248 that encrypted it, and not any other node or device. Thus, for example, if mass storage device 262 were to be removed and attempts made to read the data on device 262, such attempts would be unsuccessful. BMonitor 250 uses disk key 270 to encrypt data to be stored on mass storage device 262 regardless of the source of the data. For example, the data may come from a client device (e.g., client 102 of Fig. 1) used by a customer of the tenant, from an operations management console (e.g., console 242 of Fig. 3), etc.

Disk key 270 is generated based on symmetric keys 272 and 274. As used herein, K refers to a symmetric key, so K_L refers to a landlord symmetric key (key 272) and K_T refers to a tenant symmetric key (key 274). The individual keys 272 and 274 can be generated in any of a wide variety of conventional manners (e.g., based on a random number generator). Disk key 270 is either the K_L key alone, or alternatively is a combination of the K_L and K_T keys. In situations where the node 210 is not currently leased to a tenant, or in which the tenant has not established a K_T key, then controller 254 maintains the K_L key as disk key 270. However, in situations where the node 248 is leased to a tenant that establishes a K_T key, then disk key 270 is a combination of the K_L and K_T keys. The K_L and K_T keys can be combined in a variety of different manners, and in one implementation are combined by using one of the keys to encrypt the other key, with the resultant encrypted key being disk key 270. Thus, the data stored on mass storage device

262 is always encrypted, even if the tenant does not establish a symmetric key K_T. Additionally, in situations where the landlord and tenant are aware of their respective keys K_L and K_T, then the combination of the keys results in a key that can be used to encrypt the data so that neither the landlord nor the tenant can decrypt it individually.

In the illustrated example, a node 248 does not initially have symmetric keys K_L and K_T. When the landlord initializes the node 248, it requests a new key K_L (e.g., via cluster operations management console 240 of Fig. 3), in response to which key generator 268 generates a new key and controller 254 maintains the newly generated key as key 272. Similarly, when a tenant initially leases a node 248 there is not yet a tenant symmetric key K_T for node 248. The tenant can communicate a request for a new key K_T (e.g., via application operations management console 242 of Fig. 3), in response to which key generator 268 generates a new key and controller 254 maintains the newly generated key as key 274. Additionally, each time a new key K_T or K_L is generated, then controller 254 generates a new disk key 270.

Although only a landlord and tenant key (K_L and K_T) are illustrated in Fig. 5, alternatively additional symmetric keys (e.g., from a sub-tenant, a sub-landlord, etc.) may be combined to generate disk key 270. For example, if there are three symmetric keys, then they can be combined by encrypting a first of the keys with a second of the keys, and then encrypting the result with the third of the keys to generate disk key 270. Additional symmetric keys may be used, for example, for a sub-tenant(s).

The landlord can also request new public/private key pairs from BMonitor 250, either tenant key pairs or landlord key pairs. Requesting new key pairs can

allow, for example, the landlord to re-assign a node 248 from one tenant to another. By way of example, if a tenant no longer desires the node 248 (or does not make required lease payments for the node), then the landlord can communicate with BMonitor 250 (e.g., via console 240 of Fig. 3) to change the public/private key pairs of the tenant (thereby prohibiting any communications from the tenant from being decrypted by the BMonitor 250 because the tenant does not have the new key). Additionally, the landlord may also request a new public/private key pair for the landlord – this may be done at particular intervals or simply whenever the landlord desires a new key (e.g., for safety concerns).

In one implementation, BMonitor 250 discards both the disk key 270 and the landlord symmetric key K_L, and generates a new key K_L (and a new disk key 270) each time it generates a new landlord private key R_L. By replacing the key K_L and disk key 270 (and keeping no record of the old keys), the landlord can ensure that once it changes its key, any tenant data previously stored at the node 210 cannot be accessed. Thus, care should be taken by the landlord to generate a new public/private key pair only when the landlord wants to prevent the tenant from accessing the data previously stored at node 248.

Additionally, BMonitor 250 may also replace both the disk key 270 and the tenant symmetric key K_T, with a newly generated key K_T (and a new disk key 270) each time it generates a new tenant private key R_T. This allows the tenant to increase the security of the data being stored at the node 248 because it can change how that data is encrypted as it desires. However, as BMonitor 250 discards the previous key K_T and disk key 270, care should be exercised by the tenant to request a new tenant private key R_T only when the data previously stored at node 210 is no longer needed (e.g., has been backed up elsewhere).

It should be noted that different nodes 248 will typically have different keys (keys 264, 266, and 270). Alternatively, attempts may be made to have multiple nodes use the same key (e.g., key 270). However, in such situations care should be taken to ensure that any communication of the keys (e.g., between nodes 248) is done in a secure manner so that the security is not compromised. For example, additional public/private key pairs may be used by BMonitors 250 of two nodes 248 to securely communicate information between one another.

A leased hardware environment having guaranteed and enforced rights can thus be established. Landlords can lease nodes to multiple different tenants and establish boundaries that prevent nodes leased by different tenants from communicating with one another. Tenants can be assured that nodes they lease are accessible for management only to them, not to others, and that data is stored at the nodes securely so that no one else can access it (even if the tenant leaves or reduces its hardware usages). Furthermore, landlords and tenants are both assured that the landlord can move equipment, change which nodes are assigned to individuals, remove hardware (e.g., mass storage devices), etc. without compromising the secure storage of data by any of the tenants.

Fig. 6 is a flowchart illustrating an exemplary process for encryption key generation and distribution in accordance with certain embodiments of the invention. Initially, the computer (e.g., a node 248 of Fig. 5) identifies public/private key pairs for both the landlord and the tenant (act 280). This identification can be accessing previously generated key pairs, or alternatively generating a new key pair by the computer itself. The computer keeps both the landlord private key from the landlord key pair and the tenant private key from the tenant key pair secret, but forwards the landlord public key from the landlord key

pair and the tenant public key from the tenant key pair to the landlord (act 282). In the illustrated example, the landlord is represented by cluster operations management console 240 of Fig. 3, although alternatively other devices or entities could represent the landlord.

The landlord then forwards the tenant public key to the tenant (act 284). In the illustrated example, the tenant is represented by application operations management console 242 of Fig. 3, although alternatively other devices or entities could represent the tenant. The tenant then communicates with the computer to generate a new tenant key pair (act 286). The computer keeps the tenant private key from the new key pair secret and forwards the tenant public key from the new key pair to the tenant (act 288). The tenant is then able to communicate secure messages (e.g., data, instructions, requests, etc.) to the computer using the new tenant public key (act 290), while the landlord is able to communicate secure messages to the computer using the landlord public key (act 292).

Fig. 7 is a flowchart illustrating an exemplary process for the operation of a cluster operations management console in accordance with certain embodiments of the invention. The process of Fig. 7 is implemented by a cluster operations management console at a co-location facility, and may be performed in software.

Initially, the cluster operations management console configures the nodes in the server cluster with the boundaries (if any) of the server cluster (act 300). This configuration is accomplished by the cluster operations management console communicating filters to the nodes in the server cluster(s).

Hardware operations within a server cluster are then continually monitored for a hardware failure (acts 302 and 304). Once a hardware failure is detected, corrective action is taken (act 306) and monitoring of the hardware operation

continues. Any of a wide variety of corrective action can be taken, as discussed above. Note that, based on the corrective action (or at other times), the nodes may be re-configured with new cluster boundaries (act 300).

Fig. 8 is a flowchart illustrating an exemplary process for the operation of an application operations management console in accordance with certain embodiments of the invention. The process of Fig. 8 is implemented by an application operations management console located remotely from the co-location facility, and may be performed in software.

Initially, the application operations management console configures the nodes in the server cluster with sub-boundaries (if any) of the server cluster (act 320). This configuration is accomplished by the application operations management console communicating filters to the nodes in the server cluster.

Software operations within the server cluster are then continually monitored until a software failure is detected (acts 322 and 324). This software failure could be failure of a particular software engine (e.g., the engine fails, but the other engines are still running), or alternatively failure of the entire node (e.g., the entire node is hung). Once a software failure is detected, corrective action is taken (act 326) and monitoring of the software operation continues. Any of a wide variety of corrective action can be taken, as discussed above. Note that, based on the corrective action (or at any other time during operation), the server computer may be re-configured with new sub-boundaries (act 320).

Conclusion

Although the description above uses language that is specific to structural features and/or methodological acts, it is to be understood that the invention

defined in the appended claims is not limited to the specific features or acts described. Rather, the specific features and acts are disclosed as exemplary forms of implementing the invention.